FULL SAIL UNIVERSITY

Data Structures and Algorithms

Lab 1: DynArray

Overview

DynArray is analogous to the the vector class found in the STL (Standard Template Library). They include functionality such as **random access**, **resizeability**, and **adding/removing elements**. They have the ability to be treated like standard arrays, but have quite a bit of additional functionality that makes them more flexible than arrays.

Your task in this lab (and all subsequent ones) is to implement all of the provided methods, and pass all of the supplied unit tests. Be sure to review the class data members, as they will be used throughout the various methods. Also keep in mind the signatures of each method, in case they require a return.

The order of the unit tests is the preferred order of completing the methods. This provides a path of steadily ramping difficulty and/or the most opportunities for code reuse. This may mean that you will jump around in the file and not write the methods in the order they are declared in the class. Also keep in mind that the unit tests are independent of each other. If you get stuck at any point, turn off the unit test and move on to the next one.

Things to Review

- Arrays
- Dynamic memory
- Class data members
- Rule of 3

New Topics

- Template classes

Data Members

mArray	The actual array. This is a pointer, so that it be dynamically allocated to the appropriate size.
mCapacity	The number of total elements in the array.
mSize	The number of elements in use (storing valid data).

Methods

Constructor

- Can be called in one of two ways
- If 0 is passed in:
    - Set data members to values representing that no memory is allocated, and there is no space to store anything
- Otherwise:
    - Dynamically allocate an array of the supplied number of elements, and set your other data members to reflect that there is space to store data
- *Remember that C++ does not automatically initialize variables*

[] (Array Subscript) operator

- Think of this like an accessor for a single element, and return the data from the array, supplied at the requested index
- There should be no error-checking, as the std::vector class does not do this either

Size

- Accessor for the size data member

Capacity

- Accessor for the capacity data member

Reserve

- Resizes the array data member, based on the parameter. There are several cases that need to be handled, depending on the value passed in, as well as the current state of the class object.
- If 0 is passed in:
    - If the capacity is currently 0, set the capacity to 1
    - If the capacity is not 0, double the capacity
- If a non-zero value is passed in:
    - If the value is larger than the current capacity, set the capacity to that value
    - Otherwise, do nothing
- Dynamically allocate a temporary array of the new capacity, and copy over any data that may be in the "smaller" array
- Free up the memory of the data member
- Re-assign the data member to point to the "larger" array

Append

- Stores a value in the next empty element, and updates the size
- If there is not room to store any additional data, make room by doubling the capacity
    - *This can be done with a method you've already written*

Clear

- Cleans up all dynamically allocated memory
- Resets the object to its default state (what the data members would be after the default constructor is called)

Destructor

- Cleans up all dynamically allocated memory *(There's a method that does this)*

Assignment Operator

- Assigns all values to match those of the object passed in
- Clean up existing memory before the deep copy *(There's a method that does this)*
- Shallow copies – size and capacity
- Deep copies – array

Copy Constructor

- Creates a copy of the object passed in
- Shallow copies – size and capacity
- Deep copies – array
- *Remember that C++ does not automatically initialize data members*